Reverse Engineering of Generative Models: Inferring Model Hyperparameters from Generated Images

Vishal Asnani , Xi Yin, Tal Hassner, Xiaoming Liu

Abstract—State-of-the-art (SOTA) Generative Models (GMs) can synthesize photo-realistic images that are hard for humans to distinguish from genuine photos. Identifying and understanding manipulated media are crucial to mitigate the social concerns on the potential misuse of GMs. We propose to perform reverse engineering of GMs to infer model hyperparameters from the images generated by these models. We define a novel problem, "model parsing", as estimating GM network architectures and training loss functions by examining their generated images – a task seemingly impossible for human beings. To tackle this problem, we propose a framework with two components: a Fingerprint Estimation Network (FEN), which estimates a GM fingerprint from a generated image by training with four constraints to encourage the fingerprint to have desired properties, and a Parsing Network (PN), which predicts network architecture and loss functions from the estimated fingerprints. To evaluate our approach, we collect a fake image dataset with 100K images generated by 116 different GMs. Extensive experiments show encouraging results in parsing the hyperparameters of the unseen models. Finally, our fingerprint estimation can be leveraged for deepfake detection and image attribution, as we show by reporting SOTA results on both the deepfake detection (Celeb-DF) and image attribution benchmarks.

Index Terms—Reverse Engineering, Fingerprint Estimation, Generative Models, Deepfake Detection, Image Attribution

1 Introduction

Image generation techniques have improved significantly in recent years, especially after the breakthrough of Generative Adversarial Networks (GANs) [1]. Many Generative Models (GMs), including both GAN and Variational Autoencoder (VAE) [2], [3], [4], [5], [6], [7], [8], can generate photo-realistic images that are hard for humans to distinguish from genuine photos. This photo-realism, however, raises increasing concerns for the potential misuse of these models, *e.g.*, by launching coordinated misinformation attack [9], [10]. As a result, deepfake detection [11], [12], [13], [14], [15], [16] has recently attracted growing attention. Going beyond the binary genuine *vs*. fake classification as in deepfake detection, Yu *et al*. [17] proposed source model classification given a generated image. This *image attribution* problem assumes a *closed set* of GMs, used in both training and testing.

It is desirable to generalize image attribution to open-set recognition, *i.e.*, classify an image generated by GMs which were *not* seen during training. However, one may wonder what else we can do beyond recognizing a GM as an *unseen* or *new* model. Can we know more about how this new GM was designed? How its architecture differs from known GMs in the training set? Answering these questions is valuable when we, as defenders, strive to understand the source of images generated by malicious attackers or identify coordinated misinformation attacks which use the same GM. We view this as the grand challenge of reverse engineering of GMs.

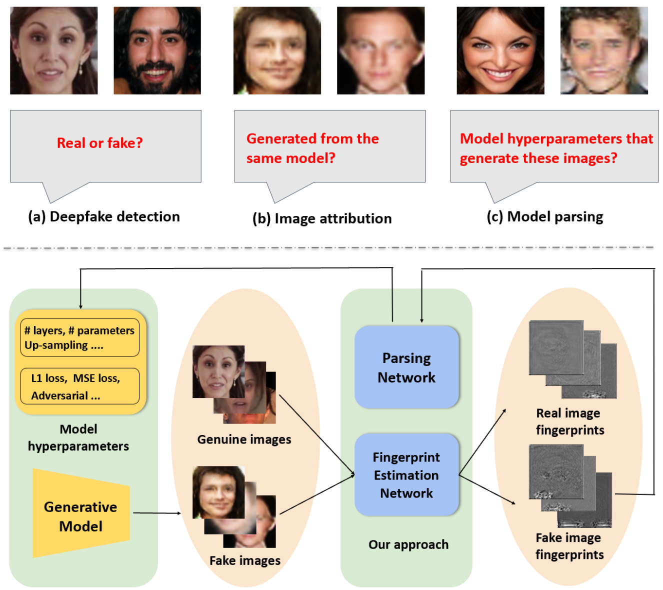


Fig. 1: Top: Three increasingly difficult tasks: (a) *deepfake detection* classifies an image as genuine or fake; (b) *image attribution* predicts which of a closed set of GMs generated a fake image; and (c) *model parsing*, proposed here, infers hyperparameters of the GM used to generate an image, for those models unseen during training. Bottom: We present a framework for model parsing, which can also be applied to simpler tasks of deepfake detection and image attribution.

While image attribution of GMs is both exciting and challenging, our work aims to take one step further with the following observation. When different GMs are designed, they mainly differ in their model hyperparameters, including the network architectures (*e.g.*, the number of layers/blocks, the type of normalization) and training loss functions. If we could map the generated images to the embedding space of the model hyperparameters used to generate them, there is a potential to tackle a new problem we

Vishal Asnani and Xiaoming Liu are with the Department of Computer Science and Engineering at Michigan State University. Xi Yin and Tal Hassner are with Meta AI. All data, experiments, and code were collected, performed, and developed at Michigan State University.

TABLE 1: Comparison of our approach with prior works on reverse engineering of models, fingerprint estimation and deepfake detection. We compare on the basis of input and output of methods, whether the testing is done on multiple unseen GMs and whether the testing is done on multiple datasets. [KEYS: R.E.: reverse engineering, I.A.: image attribution, D.D.: deepfake detection, Fing. est.: fingerprint estimation, mul.: multiple, un.: unknown, N.A.: network architecture, L.F.: Loss function, para.: parameters, sup.: supervised, unsup.: unsupervised]

Method (Year)	Purpose	Input	Output	Fing. est.	Test on mul. GMs	Test on un. GMs	Test on mul. data
[18] (2016)	R.E.	Attack on models	Training data	✗	✗	✗	✗
[19] (2018)	R.E.	Input-output images	N.A. para.	✗	✗	✗	✗
[20] (2018)	R.E.	Memory access patterns	Model weights	✗	✗	✗	✗
[21] (2018)	R.E.	Electromagnetic emanations	N.A. para.	✗	✗	✗	✗
[22] (2019)	I.A.	Image	✗	Sup.	✔	✗	✔
[17] (2019)	I.A.	Image	✗	Sup.	✔	✔	✔
[23] (2020)	I.A.	Image	✗	Sup.	✔	✗	✔
[24] (2019)	I.A.	Image	✗	Sup.	✔	✗	✔
[11] (2019)	D.D.	Image	✗	✗	✗	✗	✔
[13] (2020)	D.D.	Image	✗	✗	✗	✗	✔
[12] (2019)	D.D.	Image	✗	✗	✗	✗	✔
[14] (2019)	D.D.	Image	✗	✗	✗	✗	✔
[15] (2020)	D.D.	Image	✗	✗	✗	✗	✔
[16] (2020)	D.D.	Image	✗	✗	✗	✗	✔
[25] (2020)	D.D.	Image	✗	✗	✗	✗	✔
[26] (2021)	D.D.	Image	✗	✗	✗	✗	✔
Ours (2022)	R.E., I.A.,D.D.	Image	N.A. & L.F. para.	Unsup.	✔	✔	✔

termed as *model parsing*, *i.e.*, estimating hyperparameters of an *unseen* GM from only its generated image (Figure 1). Reverse engineering machine learning models has been done before by relying on a model's input and output [18], [19], or accessing the hardware usage during inference [20], [21]. To the best of our knowledge, however, reverse engineering has not been explored for GMs, especially with only generated images as input.

There are many publicly available GMs that generate images of diverse contents, including faces, digits, and generic scenes. To improve the generalization of model parsing, we collect a large-scale fake image dataset with various contents so that our framework is not specific to a particular content. It consists of images generated from 116 CNN-based GMs, including 81 GANs, 13 VAEs, 6 Adversarial Attack models (AAs), 11 Auto-Regressive models (ARs) and 5 Normalizing Flow models (NFs). While GANs or VAEs generate an image by feeding a genuine image or latent code to the network, AAs modify a genuine image based on its objectives via back-propagation. ARs generate each pixel of a fake image sequentially, and NFs generate images via a flow-based function. Despite such differences, we call all these models as GMs for simplicity. For each GM, our dataset includes 1,000 generated images. We use each model's hyperparameters, including network architecture parameters and training loss types, as the ground-truth for model parsing training. We propose a framework to peek inside the black boxes of these GMs by estimating their hyperparameters from the generated images. Unlike the closed-set setting in [17], we venture into quantifying the generalization ability of our method in parsing *unseen* GMs.

Our framework consists of two components (Figure 1, bottom). A *Fingerprint Estimation Network* (FEN) infers the subtle yet unique patterns left by GMs on their generated images. Image fingerprint was first applied to images captured by camera sensors [27], [28], [29], [30], [31], [32], [33] and then extended to GMs [17], [22]. We estimate fingerprints using different constraints which are based on the general properties of fingerprint, including the fingerprint magnitude, repetitive nature, frequency range and symmetrical frequency response. Different loss functions are defined to apply these constraints so that the estimated fingerprints manifest these desired properties. These constraints enable us to estimate fingerprints of GMs without ground truth.

The estimated fingerprints are discriminative and can serve as the cornerstone for subsequent tasks. The second part of our framework is a *Parsing Network* (PN), which takes the fingerprint as input and predicts the model hyperparameters. We consider parameters representing network architectures and loss function types. For the former, we form 15 parameters and categorize them into discrete and continuous types. For the latter, we form a 10-dimensional vector where each parameter represents the usage of a particular loss function type. Classification is used for estimating discrete parameters such as the normalization type, and regression is used for continuous parameters such as the number of layers. To leverage the similarity between different GMs, we group the GMs into several clusters based on their ground-truth hyperparameters. The mean and deviation are calculated for each GM. We use two different parsers: cluster parser and instance parser to predict the mean and deviation of these parameters, which are then combined as the final predictions.

Among the 116 GMs in our collected dataset, there are 47 models for face generation and 69 for non-face image generation. We partition all GMs into two categories: face *vs*. non-face. We carefully curate four evaluation sets for face and non-face categories respectively, where every set well represents the GM population. Cross-validation is used in our experiments. In addition to model parsing, our FEN can be used for deepfake detection and image attribution. For both tasks, we add a shallow network that inputs the estimated fingerprint and performs binary (deepfake detection) or multi-class classification (image attribution). Although our FEN is not tailored for these tasks, we still achieve state-of-the-art (SOTA) performance, indicating the superior generalization ability of our fingerprint estimation. Finally, in coordinated misinformation attack, attackers may use the same GM to generate multiple fake images. To detect such attacks, we also define a new task to evaluate how well our model parsing results can be used to determine if two fake images are generated from the same GM.

In summary, this paper makes the following contributions.

- We are the first to go beyond model classification by formulating a novel problem of model parsing for GMs.
- We propose a novel framework with fingerprint estimation and clustering of GMs to predict the network architecture

and loss functions, given a single generated image.

- We assemble a dataset of generated images from 116 GMs, including ground-truth labels on the network architectures and loss function types.
- We show promising results for model parsing and our fingerprint estimation generalizes well to deepfake detection on the Celeb-DF benchmark [34] and image attribution [17], in both cases reporting results comparable or better than existing SOTA [15], [17]. The parsed model parameters can also be used in detecting coordinated misinformation attacks.

2 Related Work

Reverse engineering of models. There is a growing area of interest in reverse engineering the hyperparameters of machine learning models, with two types of approaches. First, some methods treat a model as a black box API by examining its input and output pairs. For example, Tramer *et al.* [18] developed an avatar method to estimate training data and model architectures, while Oh *et al.* [19] trained a set of while-box models to estimate model hyperparameters. The second type of approach assumes that the intermediate hardware information is available during model inference. Hua *et al.* [20] estimated both the structure and the weights of a CNN model running on a hardware accelerator, by using information leaks of memory access patterns. Batina *et al.* [21] estimated the network architecture by using side-channel information such as timing and electromagnetic emanations.

Unlike prior methods which require access to the models or their inputs, our approach can reverse engineer GMs by examining *only* the images generated by these models, making it more suitable for real-world applications. We summarize our approach with previous works in Tab. 1.

Fingerprint estimation. Every acquisition device leaves a subtle but unique pattern on its captured image, due to manufacturing imperfections. Such patterns are referred to as *device fingerprints*. Device fingerprint estimation [27], [35] was extended to fingerprint estimation of GMs by Marra *et al.* [22], who showed that hand-crafted fingerprints are unique to each GM and can be used to identify an image's source. Ning *et al.* [17] extended this idea to learning-based fingerprint estimation. Both methods rely on the noise signals in the image. Others explored frequency domain information. For example, Wang *et al.* [23] showed that CNN generated images have unique patterns in their frequency domain, regarded as model fingerprints. Zhang *et al.* [24] showed that features extracted from the middle and high frequencies of the spectrum domain were useful in detecting upsampling artifacts produced by GANs.

Unlike prior methods which derive fingerprints directly from noise signals or the frequency domain, we propose several novel loss functions to learn GM fingerprints in an unsupervised manner (Tab. 1). We further show that our fingerprint estimation can generalize well to other related tasks.

Deepfake detection. Deepfake detection is a new and active field with many recent developments. Rossler *et al.* [11] evaluated different methods for detecting face and mouth replacement manipulation. Others proposed SVM classifiers on colour difference features [12]. Guarnera *et al.* [13] used Expectation Maximization [36] algorithm to extract features and convolution traces for classification. Marra *et al.* [14] proposed a multi-task incremental learning to classify new GAN generated images. Chai *et al.* [37] introduced a patch-based classifier to exaggerate regions that are more easily detectable. An attention mechanism [38] was proposed by Hao *et al.* [15] to improve the performance of deepfake detection. Masi *et al.* [25] amplifies the artifacts produced by deepfake methods to perform the detection. Nirkin *et al.* [16] seek discrepancies between face regions and their context [39] as telltale signs of manipulation. Finally, Liu [26] uses the spatial information as an additional channel for the classifier. In our work, the estimated fingerprint is fed into a classifier for genuine *vs.* fake classification.

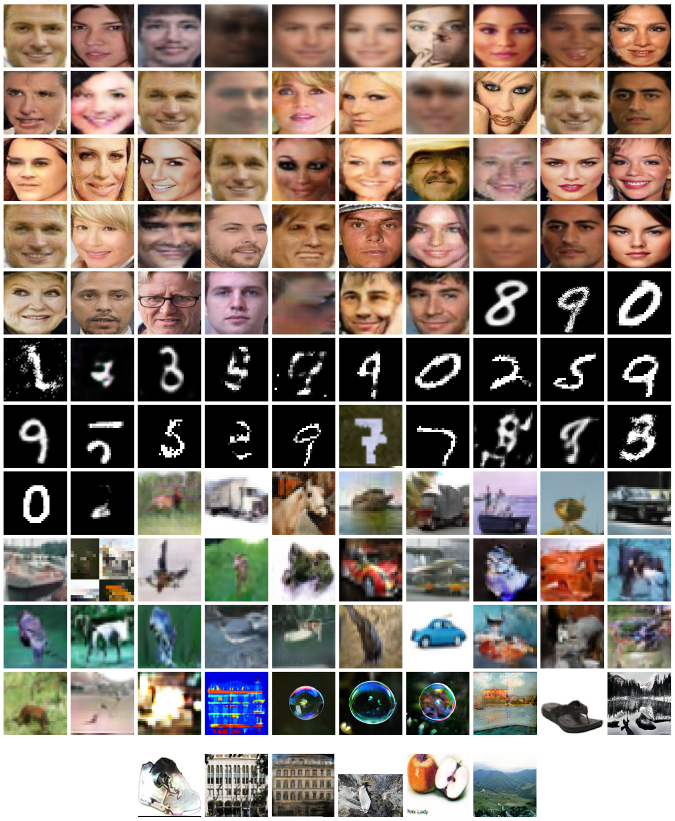


Fig. 2: Example images generated by all 116 GMs in our collected dataset (one image per model).

3 Proposed Approach

In this section, we first introduce our collected dataset in Sec. 3.1. We then present the fingerprint estimation method in Sec. 3.2 and model parsing in Sec. 3.3. Finally, we apply our estimated fingerprints to deepfake detection, image attribution, and detecting coordinated misinformation attacks, as described in Sec. 3.4.

3.1 Data collection

We make the first attempt to study the model parsing problem. Since data drives research, it is essential to collect a dataset for our new research problem. Given the large number of GMs published in recent years [40], [41], we consider a few factors while deciding which GMs to be included in our dataset. First of all, since it is desirable to study if model parsing is content-dependent, we hope to collect GMs with as diverse content as possible, such as the face, digits, and generic scenes. Secondly, we give preference to GMs where either the authors have publicly released pre-trained models, generated images, or the training script. Third,

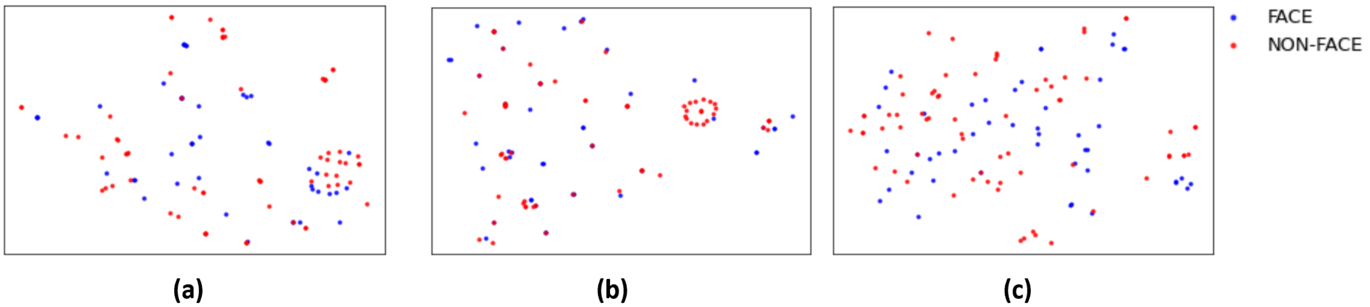


Fig. 3: t-SNE visualization for ground-truth vectors for (a) network architecture, (b) loss function and (c) network architecture and loss function combined. The ground-truth vectors are fairly distributed across the embedding space regardless of the face/non-face data.

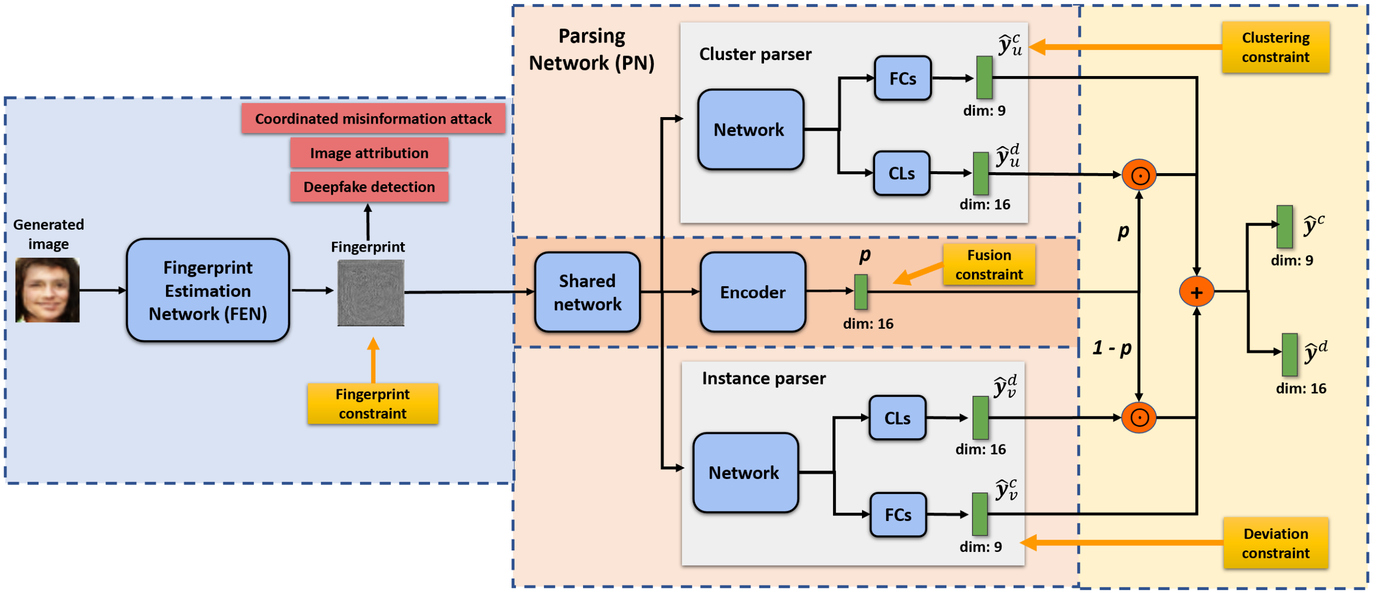


Fig. 4: Our framework includes two components: 1) the FEN is trained with four objectives for fingerprint estimation; and 2) the PN consists of a shared network, two parsers to estimate mean and deviation for each parameter, an encoder to estimate fusion parameter, fully connected layers (FCs) for continuous type parameters and separate classifiers (CLs) for discrete type parameters in network architecture and loss function prediction. Blue boxes denote trainable components; green boxes denote feature vectors; orange boxes denote loss functions; red boxes denote other tasks our framework can handle; black arrows denote data flow; orange arrows denote loss supervisions. Best viewed in color.

the network architecture of the GM should be clearly described in the respective paper.

To this end, we assemble a list of 116 publicly available GMs, including ProGan [4], StyleGAN [2], and others. A complete list is provided in the supplementary material. For each GM, we collect $1,000$ generated images. Therefore, our dataset $\mathcal{D}$ comprises of $116,000$ images. We show example images in Figure 2. These GMs were trained on datasets with various contents, such as CelebA [42], MNIST [43], CIFAR10 [44], ImageNet [45], facades [46], edges2shoes [46], and apple2oranges [46]. The dataset is available here.

We further document the model hyperparameters for each GM as reported in their papers. Specifically, we investigate two aspects: network architecture and training loss functions. We form a super-set of 15 network architecture parameters (*e.g.*, number of layers, normalization type) and 10 different loss function types. We obtain a large-scale fake image dataset $\mathbb{D} = \{\mathbf{X}_i, \mathbf{y}_i^n, \mathbf{y}_i^l\}_{i=1}^N$ where $\mathbf{X}_i$ is a fake image, $\mathbf{y}_i^n \in \mathbb{R}^{15}$ and $\mathbf{y}_i^l \in \mathbb{R}^{10}$ represent the ground-truth network architecture and loss functions, respectively. We also show the t-SNE distribution for both network architecture and loss functions in Figure 3 for different types of models and datasets. We observe that the ground-truth vectors for both network architecture and loss function are evenly distributed across the space for both types of data: face and non-face.

3.2 Fingerprint estimation

We adopt a network structure similar to the DnCNN model used in [47]. As shown in Figure 4, the input to FEN is a generated image $\mathbf{X}$, and the output is a fingerprint image $\mathbf{F}$ of the same size. Motivated by prior works on physical fingerprint estimation [17], [22], [23], [24], [48], we define the following four constraints to guide our estimated fingerprints to have the desirable properties.

Magnitude loss. Fingerprints can be considered as image noise patterns with small magnitudes. Similar assumptions were made by others when estimating spoof noise for spoofed face images [48] and sensor noise for genuine images [27]. The first

TABLE 2: Hyper-parameters representing the network architectures of GMs. (KEYS: cont. int.: continuous integer.)

Parameter	Type	Range	Parameter	Type	Range	Parameter	Type	Range
# layers	cont. int.	[5, 95]	# filter	cont. int.	[0, 8365]	non-linearity type in blocks	multi-class	0, 1, 2, 3
# convolutional layers	cont. int.	[0, 92]	# parameters	cont. int.	$[0.36M, 267M]$	non-linearity type in last layer	multi-class	0, 1, 2, 3
# fully connected layers	cont. int.	[0, 40]	# blocks	cont. int.	[0, 16]	up-sampling type	binary	0, 1
# pooling layers	cont. int.	[0, 4]	# layers per block	cont. int.	[0, 9]	skip connection	binary	0, 1
# normalization layers	cont. int.	[0, 57]	normalization type	multi-class	0, 1, 2, 3	down-sampling	binary	0, 1

TABLE 3: Loss function types used by all GMs. We group the 10 loss functions into three categories. We use the binary representation to indicate presence of each loss type in training the respective GM.

Category	Loss function
Pixel-level	L_1
	L_2
	Mean squared error (MSE)
	Maximum mean discrepancy (MMD)
	Least squares (LS)
Discriminator	Wasserstein loss for GAN (WGAN)
	Kullback–Leibler (KL) divergence
	Adversarial
	Hinge
Classification	Cross-entropy (CE)

constraint is thus proposed to regularize the fingerprint image to have a low magnitude with an L_2 loss:

$$J_m = ||\mathbf{F}||_2^2. \tag{1}$$

Spectrum loss. Previous work observed that fingerprints primarily lie in the middle and high-frequency bands of an image [24]. We thus propose to minimize the low-frequency content in a fingerprint image by applying a low pass filter to its frequency domain:

$$J_s = ||\mathcal{L}(\mathcal{F}(\mathbf{F}), f)||_2^2, \tag{2}$$

where $\mathcal{F}$ is the Fourier transform, $\mathcal{L}$ is the low pass filter selecting the $f \times f$ region in the center of the 2D Fourier spectrum and making everything else zero.

Repetitive loss. Amin *et al.* [48] noted that the noise characteristics of an image are repetitive and exist everywhere in its spatial domain. Such repetitive patterns will result in a large magnitude in the high-frequency band of the fingerprint. Therefore, we propose to maximize the high-frequency information to encourage this repetitive pattern:

$$J_r = -\max\{\mathcal{H}(\mathcal{F}(\mathbf{F}), f)\}, \tag{3}$$

where $\mathcal{H}$ is a high pass filter assigning the $f \times f$ region in the center of the 2D Fourier spectrum to zero.

Energy loss. Wang *et al.* [23] showed that unique patterns exist in the Fourier spectrum of the image generated by CNN networks. These patterns have similar energy in the vertical and horizontal directions of the Fourier spectrum. Our final constraint is proposed to incorporate this observation:

$$J_e = ||\mathcal{F}(\mathbf{F}) - \mathcal{F}(\mathbf{F})^T||_2^2, \tag{4}$$

where $\mathcal{F}(\mathbf{F})^T$ is the transpose of $\mathcal{F}(\mathbf{F})$.

These constraints guide the training of our fingerprint estimation. As shown in Figure 4, the fingerprint constraint is given by:

$$J_f = \lambda_1 J_m + \lambda_2 J_s + \lambda_3 J_r + \lambda_4 J_e, \tag{5}$$

where λ_1, λ_2, λ_3, λ_4 are the loss weights for each term.

3.3 Model parsing

The estimated fingerprint is expected to capture unique patterns generated from a GM. Prior works adopted fingerprints for deepfake detection [12], [13] and image attribution [17]. However, we go beyond those efforts by parsing the hyperparameters of GMs. As shown in Figure 4, we perform prediction using two parsers, namely, cluster parser and instance parser. We combine both outputs for network architecture and loss function prediction. We will now discuss the ground truth calculation and our framework in detail.

3.3.1 Ground truth hyperparamters

Network architecture. In this work, we do not aim to recover the network parameters. The reason is that a typical deep network has millions of network parameters, which reside in a very high dimensional space and is thus hard to predict. Instead, we propose to infer the hyperparameters that define the network architecture, which are much fewer than the network parameters. Motivated by prior works in neural architecture search [49], [50], [51], we form a set of 15 network architecture parameters covering various aspects of architectures. As shown in Tab. 2, these parameters fall into different data types and have different ranges. We further split the network architecture parameters $\mathbf{y}^n$ into two parts: $\mathbf{y}^{n_c} \in \mathbb{R}^9$ for continuous data type and $\mathbf{y}^{n_d} \in \mathbb{R}^6$ for discrete data type.

Loss function. In addition to the network architectures, the learned network parameters of trained GM can also impact the fingerprints left on the generated images. These network parameters are determined mainly by the training data and the loss functions used to train these models. We, therefore, explore the possibility of also predicting the training loss functions from the estimated fingerprints. The 116 GMs were trained with 10 types of loss functions as shown in Tab. 3. For each model, we compose a ground-truth vector $\mathbf{y}^l \in \mathbb{R}^{10}$, where each element is a binary value indicating whether the corresponding loss is used or not in training this model.

Our framework parses two types of hyperparameters: continuous and discrete. The former includes the continuous network architecture parameters. The latter includes discrete network architecture parameters and loss function parameters. For clarity, we group these parameters into continuous and discrete types in the remaining of this section to describe the model parsing objectives. We use $\mathbf{y}^c$ and $\mathbf{y}^d$ to denote continuous and discrete parameters respectively.

3.3.2 Cluster parser prediction

We have observed that directly estimating the hyperparameters independently for each GM yields inferior results. In fact, some of the GMs in our dataset have similar network architectures and/or loss functions. It is intuitive to leverage the similarities among different GMs for better hyperparameter estimation. To do this, we perform k-means clustering to group all GMs into different clusters, as shown in Figure 5. Then we propose to perform

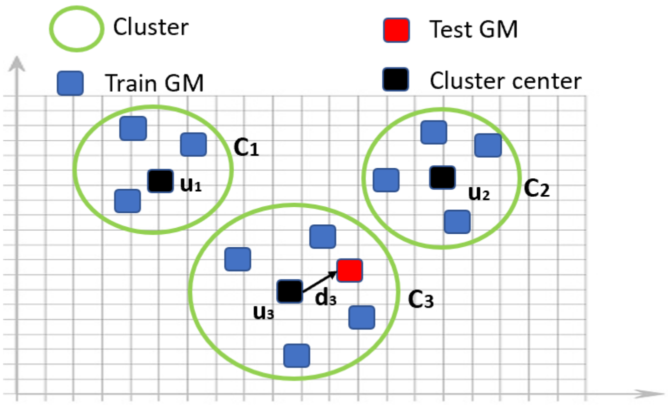


Fig. 5: The idea of grouping various GMs into different clusters. For the test GM, we estimate its cluster mean and the deviation from that mean to predict network architecture and loss function type.

cluster-level coarse prediction and GM-level fine prediction, which are subsequently combined to obtain the final prediction results.

As we aim to estimate the parameters for network architecture and loss function, it is intuitive to combine them to perform grouping. Thus, we concatenate the ground truth network architecture parameters $\mathbf{y}^n$ and loss function parameters $\mathbf{y}^l$, denoted as $\mathbf{y}^{nl}$. We use these ground truth vectors to perform k-means clustering to find the optimal k-clusters in the dataset $\mathcal{D} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, ...\boldsymbol{C}_k\}$. Our clustering objective can be written as:

$$\arg\min_{\mathcal{D}} \sum_{i=1}^{k} \sum_{\mathbf{y}_j^{nl}\in\boldsymbol{C}_i} ||\mathbf{y}_j^{nl} - \mu_i||^2, \tag{6}$$

where μ_i is the mean of the ground truth of the GMs in $\boldsymbol{C}_i$.

Our dataset comprises different kinds of GMs, namely GANs, VAEs, AAs, ARs, and NFs. We perform clustering after separating the training data into different kinds of GMs. This is done to ensure that each cluster would belong to one particular kind of GM. Next, we select the value of k *i.e.*, the number of clusters, using the elbow method adopted by previous works [52], [53]. After determining the clusters comprising of similar GMs, we estimate the ground truth $\mathbf{y}_u$ to represent the respective cluster. We estimate this cluster ground truth using different ways for continuous and discrete parameters. For the former, we take the average of each parameter using the ground truth for all GMs in the respective cluster. For the latter, we perform majority voting for every parameter to find the most common class across all GMs in the cluster.

We use different loss functions to perform cluster-level prediction. For continuous parameters, we perform regression for parameter estimation. As these parameters are in different ranges, we further perform a min-max normalization to bring all parameters to the range of [0, 1]. An L_2 loss is used to estimate the prediction error:

$$J_u^c = ||\hat{\mathbf{y}}_u^c - \mathbf{y}_u^c||_2^2, \tag{7}$$

where $\hat{\mathbf{y}}_u^c$ is the cluster mean prediction and $\mathbf{y}_u^c$ is the normalized ground-truth cluster mean.

For discrete parameters, the prediction is made via individual classifiers. Specifically, we train $M = 16$ classifiers (6 for network architecture and 10 for loss function parameters), one for each discrete parameter. The loss term for discrete parameters cluster-prediction is defined as:

$$J_u^d = -\sum_{m=1}^{M} \text{sum}(\mathbf{y}_{u_m}^d \odot \log(\mathcal{S}(\hat{\mathbf{y}}_{u_m}^d))), \tag{8}$$

where $\mathbf{y}_{u_m}^d$ is the ground-truth one-hot vector for the respective class in the m-th discrete type parameter, $\hat{\mathbf{y}}_{u_m}^d$ are the class logits, $\mathcal{S}$ is the Softmax function that maps the class logits into the range of $[0, 1]$, $\odot$ is the element-wise multiplication, and $\text{sum}()$ computes the summation of a vector's elements.

As shown in Figure 4, the clustering constraint is given by:

$$J_u = \gamma_1 J_u^c + \gamma_2 J_u^d, \tag{9}$$

where γ_1 and γ_2 are the loss weights for each term.

3.3.3 *Instance parser prediction*

The cluster parser performs coarse-level prediction. To obtain a more fine-level prediction, we use an instance parser to estimate a GM-level prediction, which ignores any similarity among GMs. This parser aims to predict the deviation of every parameter from the coarse-level prediction. The ground truth deviation vector $\mathbf{y}_v$ can be estimated in different ways for two types of parameters. For continuous type parameters, the deviation can be the difference between the ground truth of the GM and the ground truth of the cluster the GM was assigned. However, in the case of discrete parameters, the actual ground truth class for the parameters can act as the deviation from the most common class estimated in cluster ground truth. We use different loss functions to perform deviation-level prediction. Specifically, we use an L_2 loss to estimate the prediction error for continuous parameters:

$$J_v^c = ||\hat{\mathbf{y}}_v^c - \mathbf{y}_v^c||_2^2, \tag{10}$$

where $\hat{\mathbf{y}}_v^c$ is the deviation prediction and $\mathbf{y}_v^c$ is the deviation ground-truth of continuous data type.

We have noticed the class distribution for some discrete parameters is imbalanced. Therefore, we apply the weighted cross-entropy loss for every parameter to handle this challenge. We train $M = 16$ classifiers, one for each of the discrete parameters. For the m-th classifier with N_m classes ($N_m = 2$ or 4 in our case), we calculate a loss weight for each class as $w_m^i = \frac{N}{N_m^i}$ where N_m^i is the number of training examples for the ith class of m-th classifier, and N is the number of total training examples. As a result, the class with more examples is down-weighted, and the class with fewer examples is up-weighted to overcome the imbalance issue, which will be empirically demonstrated in Figure. 9. The loss term for discrete parameters deviation-prediction is defined as:

$$J_v^d = -\sum_{m=1}^{M} \text{sum}(\mathbf{w}_m \odot \mathbf{y}_{v_m}^d \odot \log(\mathcal{S}(\hat{\mathbf{y}}_{v_m}^d))), \tag{11}$$

where $\mathbf{y}_{v_m}^d$ is the ground-truth one-hot deviation vector for the m-th classifier, $\mathbf{w}_m$ is a weight vector for all classes in the m-th classifier and $\hat{\mathbf{y}}_{v_m}^d$ are the class logits.

As shown in Figure 4, the deviation constraint is given by:

$$J_v = \gamma_3 J_v^c + \gamma_4 J_v^d. \tag{12}$$

where γ_3 and γ_4 are the loss weights for each term.

3.3.4 Combining predictions

We use a cluster parser to perform a coarse-level mean prediction and an instance parser to predict a deviation prediction for each GM. The final prediction of our framework, *i.e.*, the prediction at the fine-level is the combination of the outputs of these two parsers. For continuous parameters, we perform the element-wise addition of the coarse-level mean and deviation prediction:

$$\hat{\mathbf{y}}^c = \hat{\mathbf{y}}^c_u + \hat{\mathbf{y}}^c_v, \tag{13}$$

For discrete parameters, we have observed that element-wise addition of the logits for every classifier in both parsers didn't perform well. Therefore, to integrate the outputs, we train an encoder network to predict a fusion parameter $\hat{p}^d \in [0, 1]$ for each classifier. For any parameter, the value of the fusion parameter is 1 if the cluster class is the same as the GM class, encouraging the parsing network to give importance to the cluster parser output. The value of the fusion parameter is 0 if the GM class is different from the cluster class. Therefore, for m-th classifier, the training of the model is supervised by the ground truth p^d_m as defined below:

$$p^d_m = \begin{cases} 1, & \mathbf{y}^d_{u_m} = \mathbf{y}^d_{v_m} \\ 0, & \mathbf{y}^d_{u_m} \neq \mathbf{y}^d_{v_m}. \end{cases} \tag{14}$$

To train our encoder, we use the ground truth fusion parameter $\mathbf{p}^d$ which is the concatenation for all parameters. The training is done via cross-entropy loss as shown below:

$$J_p = -\sum_{m=1}^{M} (p^d_m \log(\mathcal{G}(\hat{p}^d_m)) + (1 - p^d_m) \log(1 - \mathcal{G}(\hat{p}^d_m))). \tag{15}$$

where $\mathcal{G}$ is the Sigmoid function that maps the class logits into the range of $[0, 1]$.

As shown in Figure 4 for discrete parameters, the final prediction is given by:

$$\hat{\mathbf{y}}^d = \hat{\mathbf{p}}^d \odot \hat{\mathbf{y}}^d_u + (\mathbf{1} - \hat{\mathbf{p}}^d) \odot \hat{\mathbf{y}}^d_v. \tag{16}$$

The overall loss function for model parsing is given by:

$$J = J_f + J_u + J_v + \gamma_5 J_p. \tag{17}$$

where γ_5 is the loss weight for fusion constraint. Our framework is trained end-to-end with fingerprint estimation (Eqn. 5) and model parsing (Eqn. 17).

3.4 Other applications

In addition to model parsing, our fingerprint estimation can be easily leveraged for other applications such as detecting coordinated misinformation attacks, deepfake detection and image attribution.

Coordinated misinformation attack. In coordinated misinformation attacks, the attackers often use the same model to generate multiple fake images. One way to detect such attacks is to classify whether two fake images are generated from the same GM, despite that this GM might be unseen to the classifier. This task is not straightforward to perform by prior works. However, given the ability of our model parsing, this is the ideal task that we can contribute. To perform this binary classification task, we use the parsed network architecture and loss function parameters to calculate the similarity score between two test images. We calculate the cosine similarity for continuous type parameters and fraction of the number of parameters having same class for discrete type. Both cosine similarity and fraction of parameters are averaged to get the similarity score. Comparing the cosine similarity with a threshold will lead to the binary classification decision of whether two images come from the same GM or not.

Deepfake detection. We consider the binary classification of an image as either genuine or fake. We add a shallow network on the generated fingerprint to predict the probabilities of being genuine or fake. The shallow network consists of five convolution layers and two fully connected layers. Both genuine and fake face images are used for training. Both FEN and the shallow network are trained end-to-end with the proposed fingerprint constraints (Eqn. 5) and a cross-entropy loss for genuine *vs*. fake classification. Note that the fingerprint constraints (Eqn. 5) are not applied to the genuine input face images.

Image attribution. We aim to learn a mapping from a given image to the model that generated it if it is fake or classified as genuine otherwise. All models are known during training. We solve image attribution as a closed-set classification problem. Similar to deepfake detection, we add a shallow network on the generated fingerprint for model classification with the cross-entropy loss. The shallow network consists of two convolutional layers and two fully connected layers.

4 Experiments

4.1 Settings

Dataset. As described in Sec. 3.1, we have collected a fake image dataset consisting of $116K$ images from 116 GMs ($1K$ images per model) for model parsing experiments. These models can be split into two parts: 47 face models and 69 non-face models. Instead of performing one split of training and testing sets, we carefully construct four different splits with a focus on curating well-represented test sets. Specifically, each testing set includes six GANs, two VAEs, two ARs, one AA and one NF model. We perform cross-validation to train on 104 models and evaluate on the remaining 12 models in testing sets. The performance is averaged across four testing sets.

For deepfake detection experiments, we conduct experiments on the recently released Celeb-DF dataset [34], consisting of 590 real and 5,639 fake videos. For image attribution experiments, a source database with genuine images needs to be selected, from which the fake images can be generated by various GAN models. We select two source datasets: CelebA [34] and LSUN [54], for two experiments. From each source dataset, we construct a training set of $100K$ genuine and $100K$ fake face images produced by each of the same four GAN models used in Yu *et al*. [17], and a testing set with $10K$ genuine and $10K$ fake images per model.

Implementation details. Our framework is trained end-to-end with the loss functions of Eqn. 5 and Eqn. 17. The loss weights are set to make the magnitudes of all loss terms comparable: $\lambda_1 = 0.05$, $\lambda_2 = 0.001$, $\lambda_3 = 0.1$, $\lambda_4 = 1$, $\gamma_1 = 5$, $\gamma_2 = 5$, $\gamma_3 = 5$, $\gamma_4 = 5$, $\gamma_5 = 5$, $\gamma_6 = 5$, $\gamma_7 = 1$, $\gamma_8 = 1$. The value of f for spectrum loss and repetitive loss in the fingerprint estimation is set to 50. For each of the four test sets, we calculate the number of clusters k using the elbow method. We divide the data into different GM types and perform k-means clustering separately for each type. According to the sets defined in the supplementary, we obtain the value of k as 11, 11, 15, and 13. We use Adam optimizer with a learning rate of 0.0001. Our framework is trained with a batch size of 32 for 10 epochs. All the experiments are conducted using NVIDIA Tesla K80 GPUs.

Evaluation metrics. For continuous type parameters, we report the L_1 error for the regression estimation of continuous type parameters. We also report the p-value of t-test, correlation coefficient, coefficient of determination [55] and slope of the RANSAC regression line [56] to show the effectiveness of regression in our approach. For discrete type parameters, as there is imbalance in the dataset for different parameters, we compute the F1 score [57], [58] for classification performance. We also report classification accuracy for discrete-type parameters. For all cross-validation experiments, we report the averaged results across all images and all GMs.

4.2 Model parsing results

As we are the first to attempt GM parsing, there are no prior works for comparison. To provide a baseline, we, therefore, draw an analogy with the image attribution task, where each model is represented as a one-hot vector and different models have equal inter-model distances in the high-dimensional space defined by these one-hot vectors. In model parsing, we represent each model as a 25-D vector consisting of network architectures (15-D) and training loss functions (10-D). Thus, these models are not of equal distance in the 25-D space.

Based on the aforementioned observation, we define a baseline, referred to here as *random ground-truth*. Specifically, for each parameter, we shuffle the values/classes across all 116 GMs to ensure that the assigned ground-truth is different from the actual ground-truth but also preserves the actual distribution of each parameter, which means that the random ground-truth baseline is not based on random chance. These random ground-truth vectors have the same properties as our ground-truth vectors in terms of non-equal distances. But the shuffled ground truths are meaningless and are not corresponding to their true model hyperparameters. We train and test our proposed approach on this randomly shuffled ground-truth. Due to the random nature of this baseline, we perform three random shuffling and then report the average performance. We also evaluate a baseline of always predicting the mean for continuous hyperparameters, and always predicting the mode for discrete hyperparameters across the four sets. These mean/mode values of the hyperparameters are both measures of central tendency to represent the data, and they might result in a good enough performance for model parsing.

To validate the effects of our proposed fingerprint estimation constraints, we conduct an ablation study and train our framework end-to-end with only the model parsing objective in Eqn. 17. This results in the *no fingerprint* baseline. Finally, to show the importance of our clustering and deviation parser, we estimate the network architecture and loss functions using just one parser, which estimates the parameters directly instead of a mean and deviation. We refer to this as *using one parser* baseline.

Network architecture prediction. We report the results of network architecture prediction in Tab. 4 for the 4 testing sets, as defined in Sec. 4.1. Our method achieves a much lower L_1 error compared to the random ground-truth baseline for continuous type parameters and higher classification accuracy and F1 score for discrete type parameters. This result indicates that there is indeed a much stronger and generalized correlation between generated images and the embedding space of meaningful architecture hyper-parameters and loss function types, compared to a random vector of the same length and distribution. This correlation is the foundation of why model parsing of GMs can be a valid and feasible task. Our approach also outperforms the mean/mode baseline, proving that always predicting the mean of the data for continuous parameters is not good enough. Removing fingerprint estimation objectives leads to worse results showing the importance of the fingerprint estimation in model parsing. We demonstrate the effectiveness of estimating mean and deviation by evaluating the performance of using just one parser. Our method clearly outperforms the approach of using one parser.

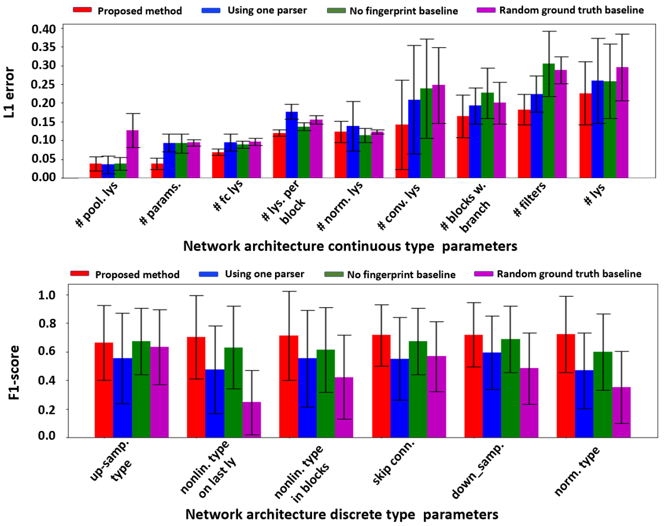


Fig. 6: L_1 error and F1 score for continuous and discrete parameters respectively of network architecture averaged across all images of all models in the 4 test sets. Not only we have better average performance, but also our standard deviations are smaller.

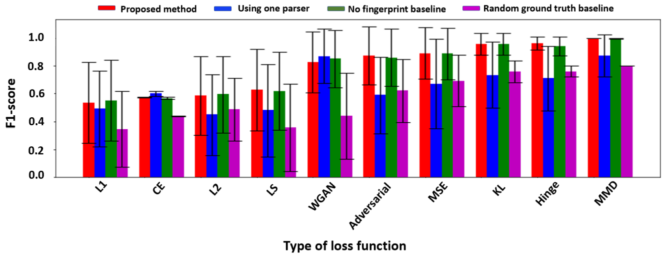


Fig. 7: F1 score for each loss function type at coarse and fine levels averaged across all images of all models in the 4 test sets. We also show the standard deviation of performance across different sets.

Figure 6 shows the detailed L_1 error and F1 score for all network architecture parameters. We observe that our method performs substantially better than the random ground-truth baseline for almost all parameters. As for the no fingerprint and using one parser baselines, our method is still better in most cases with a few parameters showing similar results. We also show the standard deviation of every estimated parameter for all the methods. Our proposed approach in general has smaller standard deviations than the two baselines. For continuous type parameters, we further show the effectiveness of regression prediction by evaluating three metrics namely, correlation coefficient, coefficient of determination and slope of RANSAC regression line. These metrics are evaluated between prediction and ground-truth. Further, we also estimate a p-value of a t-test, where the null hypothesis is as follows: the sequence of sample-wise L_1 error differences between our method and the baseline method is sampled from zero-mean Gaussian. This p-value would be estimated for every

TABLE 4: Performance of network architecture prediction. We use L_1 error, p-value, correlation coefficient, coefficient of determination and slope of RANSAC regression line for continuous type parameters. For discrete parameters, we use F1 score and classification accuracy. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples. The p-value would be estimated for every ours-baseline pair. Our method performs better for both types of variables compared to the three baselines. [KEYS: corr.: correlation, coef.: coefficient, det.: determination]

Method	Continuous type					Discrete type	
	L_1 error ↓	P-value ↓	Corr. coef. ↑	Coef. of det. ↑	Slope ↑	F1 score ↑	Accuracy ↑
Random ground-truth	$0.184 \pm 0.019/0.036$	0.006 ± 0.001	0.261 ± 0.181	0.315 ± 0.095	0.592 ± 0.041	0.529 ± 0.078	0.575 ± 0.097
Mean/mode	$0.164 \pm 0.011/0.016$	0.035 ± 0.005	0.326 ± 0.112	0.467 ± 0.015	0.632 ± 0.024	0.612 ± 0.048	0.604 ± 0.046
No fingerprint	$0.170 \pm 0.035/0.012$	0.017 ± 0.004	0.738 ± 0.014	0.605 ± 0.152	0.892 ± 0.021	0.700 ± 0.032	0.663 ± 0.104
Using one parser	$0.161 \pm 0.028/0.035$	0.032 ± 0.002	0.226 ± 0.030	0.512 ± 0.116	-0.529 ± 0.075	0.607 ± 0.034	0.593 ± 0.104
Ours	$\mathbf{0.149 \pm 0.019/0.014}$	-	$\mathbf{0.744 \pm 0.098}$	$\mathbf{0.612 \pm 0.161}$	$\mathbf{0.921 \pm 0.021}$	$\mathbf{0.718 \pm 0.036}$	$\mathbf{0.706 \pm 0.040}$

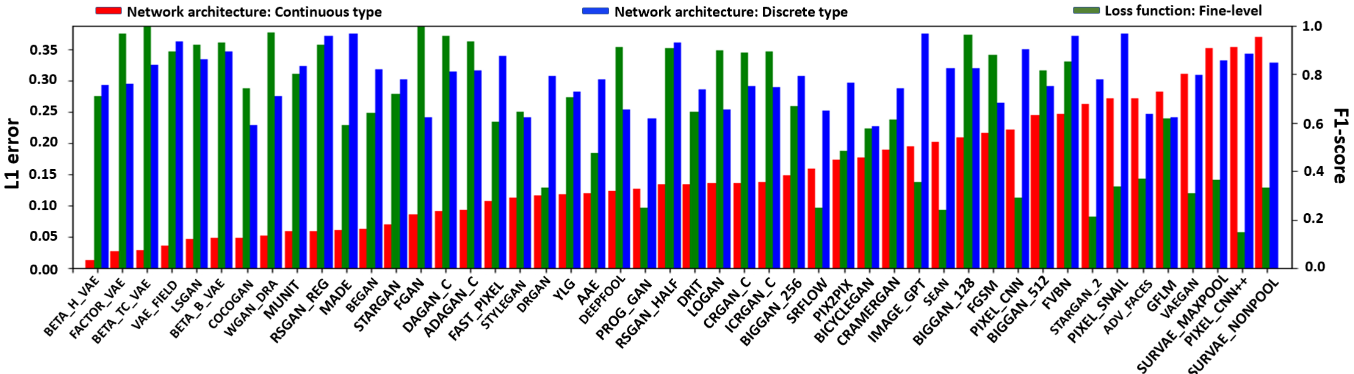


Fig. 8: Performance of all GMs in our 4 testing sets. Similar performance trends are observed for network architecture and loss functions, *i.e.*, if the L_1 error is small for continuous type parameters in network architecture, the high F1 score is also observed for discrete type parameters in network architecture and loss function. In other words, the abilities to reverse engineer the network architecture and loss function types for one GM are reasonably consistent.

TABLE 5: F1 score and classification accuracy for loss type prediction. Our method performs better than all the three baselines.

Method	Loss function prediction	
	F1 score ↑	Classification accuracy ↑
Random ground-truth	0.636 ± 0.017	0.716 ± 0.028
Mean/mode	0.751 ± 0.027	0.736 ± 0.056
No fingerprint	0.800 ± 0.116	0.763 ± 0.079
Using one parser	0.687 ± 0.036	0.633 ± 0.052
Ours	$\mathbf{0.813 \pm 0.019}$	$\mathbf{0.792 \pm 0.021}$

ours-baseline pair. We report the mean and the standard deviation across all four sets. The p-value of our approach when compared to all the three baselines is less than 0.05, thereby rejecting the null hypothesis and proving our improvement is statistically significant. For other three metrics, the values closer to 1 shows effective regression. For our method, we have slope of 0.921, correlation coefficient of 0.744 and coefficient of determination as 0.612 which shows the effectiveness of our approach. Further, our approach outperforms all the baselines for all three metrics.

Loss function prediction. We calculate the F1 score and classification accuracy for loss function parameters. The performance are shown in Tab. 5. For the random ground-truth baseline, the performance is close to a random guess. Our approach performs much better than all the baselines. Figure 7 shows the detailed F1 score for all loss function parameters. Apparently our method works better than all the baselines for almost all parameters. We also show the standard deviation of every estimated parameter for all the methods. Similar behaviour of standard deviation for different methods was observed as in the network architecture. Figure 8 provides another perspective of model parsing by showing the performance in terms of 48 unique GMs across our 4 testing sets.

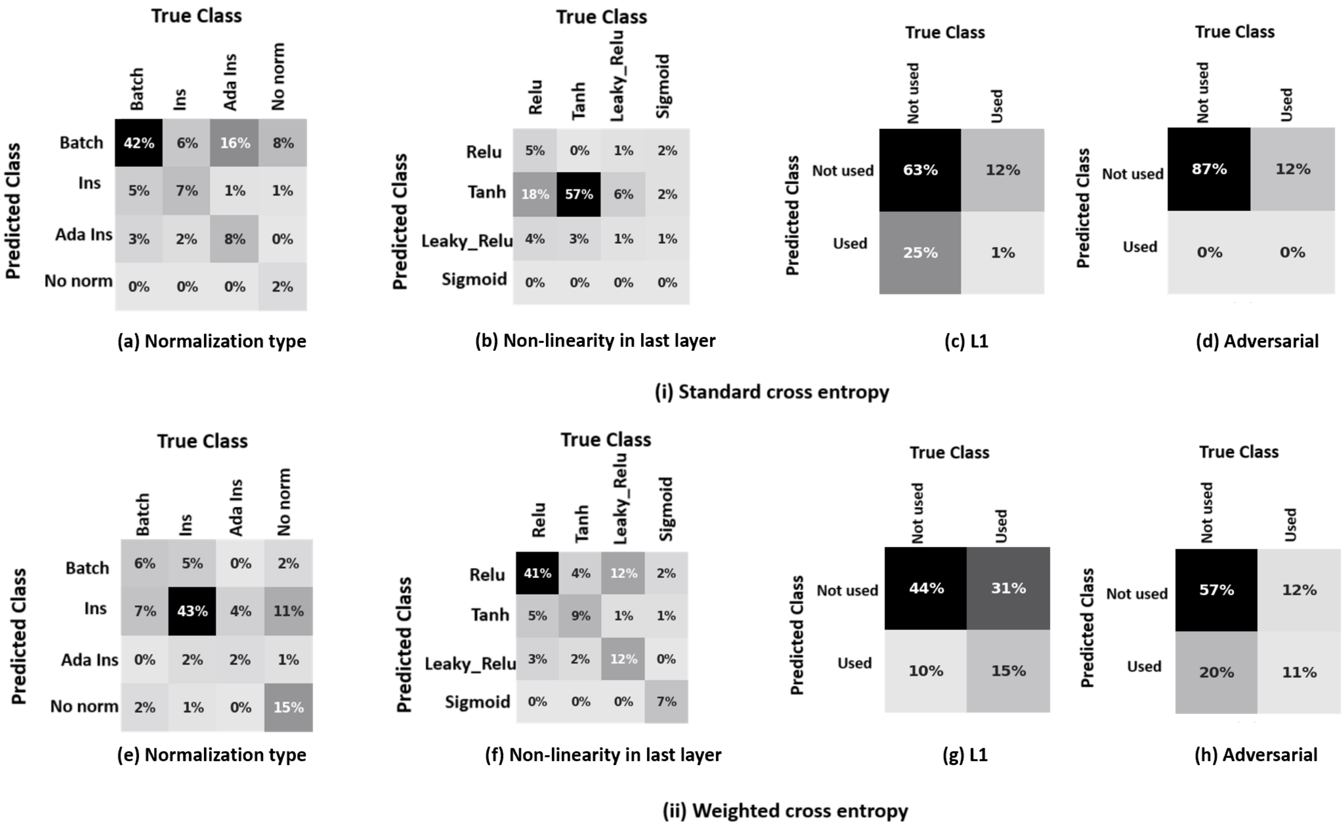
Practical Usage of Model Parsing.. As our work is the first one to propose the task of model parsing, it's beneficial to ask the question: *what is the performance desired for practical usage of model parsing in the real world?* To answer this question, we can expect that an error less than 10% can be considered useful for the practical application of model parsing. The rationale is the following. We consider two of the most similar generative models, RSGAN_HALF and RSGAN_QUAR, in our dataset. Upon further analysis, we observe that these models differ in only 2 out of 15 parameters. Therefore, we argue that an error rate below 10% is reasonable for practical purposes as this error is less than the difference between the two most similar generative models. Therefore, for the task of model parsing, we expect L_1 error of less than 0.1 and an $F1$ score of over 90% for practical usage. Our proposed approach achieves an L_1 error slightly above 10% (0.14) and an $F1$ score of 80%, both of which have reasonable margins toward the above mentioned thresholds.

4.3 Ablation study

Face *vs*. non-face GMs. Our dataset consists of 47 GMs trained on face datasets and 69 GMs trained on non-face datasets. Let's denote these GMs as face GMs and non-face GMs, respectively. All aforementioned experiments are conducted by training on 104 GMs and evaluating on 12 GMs. Here we conduct an ablation study to train and evaluate on different types of GMs. We study the performance on face and non-face testing GMs when training on three different training sets, including only face GMs, only non-face GMs and all GMs. Note that all testing GMs are

TABLE 6: Performance comparison by varying the training and testing data for face and non-face GMs. Testing performance on non-face GMs is better compared to face GMs. Training and testing on the same content produces better results than on the different contents.We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

Test GMs (# GMs)	Train GMs (# GMs)	Network architecture Continuous type L_1 error ↓	Network architecture Discrete type F1 score ↑	Loss function F1 score ↑
Face (6)	Face (41)	$0.139 \pm 0.042/0.015$	$\mathbf{0.729 \pm 0.106}$	0.788 ± 0.146
	Non-face (69)	$0.213 \pm 0.066/0.136$	0.688 ± 0.125	0.759 ± 0.100
	Full (110)	$\mathbf{0.118 \pm 0.046/0.040}$	0.712 ± 0.129	$\mathbf{0.833 \pm 0.136}$
Non-face (6)	Non-face (63)	$0.118 \pm 0.021/0.049$	0.794 ± 0.110	0.864 ± 0.094
	Face (47)	$0.125 \pm 0.031/0.028$	0.667 ± 0.099	0.858 ± 0.115
	Full (110)	$\mathbf{0.082 \pm 0.045/0.049}$	$\mathbf{0.832 \pm 0.046}$	$\mathbf{0.886 \pm 0.061}$
Random guess		0.393	0.500	0.500



(a) Normalization type (b) Non-linearity in last layer (c) L1 (d) Adversarial

(i) Standard cross entropy

(e) Normalization type (f) Non-linearity in last layer (g) L1 (h) Adversarial

(ii) Weighted cross entropy

Fig. 9: Confusion matrix in the estimation of four parameters in the network architecture and loss function. (a)-(d): Standard cross-entropy and (e)-(f): Weighted cross entropy. Weighted cross entropy handles imbalance data much better than the standard cross entropy which usually predicts one class.

excluded during training each time. We also add a baseline where both regression and classification make a random guess on their estimation.

The results are shown in Tab. 6. We have three observations. First, model parsing for non-face GMs are easier than face GMs. This might be partially due to the generally lower-quality images generated by non-face GMs compared to those by face GMs, thus more traces are remained for model parsing. Second, training and testing on the same content can generate better results than on different contents. Third, training on the full datasets improves some parameter estimation but may hurt other parameters slightly.

Weighted cross-entropy loss. As mentioned before, the ground truth of many network hyperparameters have biased distributions. For example, the "normalization type" parameter in Tab. 2 has uneven distribution among its 4 possible types. With this biased distribution, our classifier might make a constant prediction to the type with the highest probability in the ground truth, as this could minimize the loss especially for severe biasness. This degenerate classifier clearly has no value to model parsing. To address this issue, we propose to use the weighted cross-entropy loss with different loss weights for each class. These weights are calculated using the ground-truth distribution of every parameter in the full dataset. To validate if the above approach is able to remedy this issue, we compare it with the standard cross-entropy loss.

Figure 9 shows the confusion matrix for discrete type parameters in network architecture prediction and coarse/fine level parameters in loss function prediction. The rows in the confusion matrix are represented by predicted classes and columns are represented by the ground-truth classes. We clearly see that the classifier is mostly biased towards more frequent classes in all 4

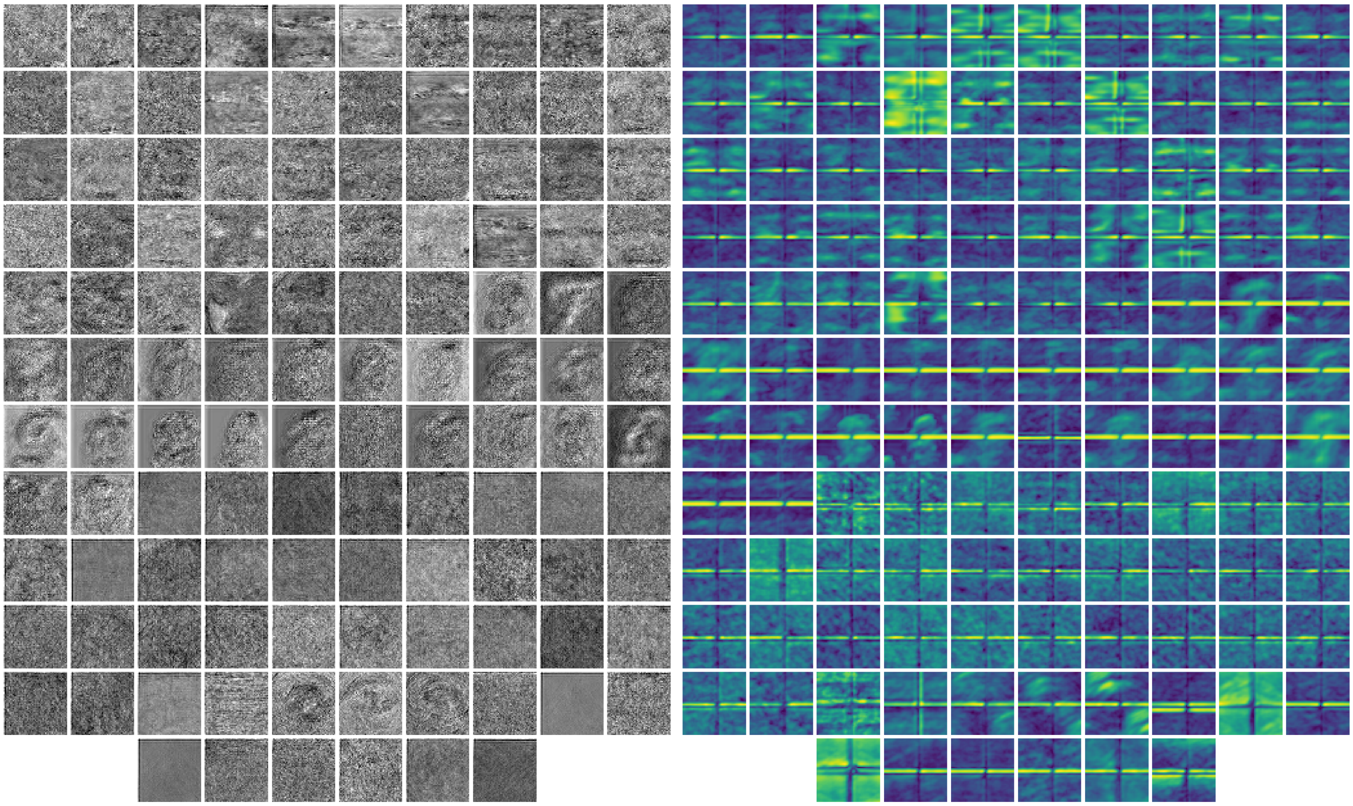


Fig. 10: Estimated fingerprints (left) and corresponding frequency spectrum (right) from one generated image of each of 116 GMs. Many frequency spectrums show distinct high-frequency signals, while some appear to be similar to each other.

TABLE 7: Ablation study of the 4 loss terms in fingerprint estimation. Removing any one loss for fingerprint estimation deteriorates the performance with the worst results in the case of removing all losses. [KEYS: fing.: fingerprint]. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

Loss removed	Network architecture		Loss function
	Continuous type L_1 error ↓	Discrete type F1 score ↑	F1 score ↑
Magnitude loss	$0.156 \pm 0.007/0.009$	0.674 ± 0.012	0.755 ± 0.046
Spectrum loss	$\mathbf{0.149 \pm 0.022/0.016}$	0.676 ± 0.034	0.786 ± 0.042
Repetitive loss	$0.150 \pm 0.018/0.026$	0.708 ± 0.031	0.794 ± 0.031
Energy loss	$0.162 \pm 0.032/0.038$	0.703 ± 0.045	0.785 ± 0.028
All (no fing.)	$0.170 \pm 0.035/0.037$	0.700 ± 0.032	0.800 ± 0.016
Nothing (ours)	$\mathbf{0.149 \pm 0.019/0.014}$	$\mathbf{0.718 \pm 0.036}$	$\mathbf{0.813 \pm 0.019}$

TABLE 8: Network architecture estimation and loss function prediction when given multiple images of one GM. Performance increases when enlarging the number of images for evaluation from 1 to 10. Performance becomes stable for more than 10 images. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

# images	Network architecture		Loss function
	Continuous type L_1 error ↓	Discrete type F1 score ↑	F1 score ↑
1	$0.215 \pm 0.054/0.067$	0.696 ± 0.089	0.798 ± 0.010
10	$0.151 \pm 0.033/0.039$	$\mathbf{0.726 \pm 0.075}$	0.793 ± 0.070
100	$\mathbf{0.145 \pm 0.032/0.036}$	0.721 ± 0.073	0.789 ± 0.071
500	$0.146 \pm 0.033/0.031$	0.720 ± 0.070	$\mathbf{0.808 \pm 0.007}$

examples, when the standard cross-entropy loss is used. However, this problem is remedied when using the weighted cross-entropy loss, and the classifiers make meaningful predictions.

Fingerprint losses. We proposed four loss terms in Sec. 3.2 to guide the training of the fingerprint estimation including magnitude loss, spectrum loss, repetitive loss and energy loss. We conduct an ablation study to demonstrate the importance of these four losses in our proposed method. This includes four experiments, each removing one of the loss terms and comparing the performance with our proposed method (remove nothing) and no fingerprint baseline (remove all). As shown in Tab. 7, removing any loss for fingerprint estimation hurts the performance. Our "no fingerprint" baseline, for which we remove all losses, performs worst of all. Therefore, each loss clearly has a positive effect on the fingerprint estimation and model parsing.

Model parsing with multiple images. We evaluate model parsing when varying the number of test images. For each GM, we randomly select 1, 10, 100, and 500 images per GM from different face GMs sets for evaluation. With multiple images per GM, we average the prediction for continuous type parameters and take majority voting for discrete type parameters and loss function parameters. We compute the L_1 error and F1 score for the continuous and discrete type parameters respectively and average the result across different sets. We repeat the above experiment multiple times, each time randomly selecting the number of images. We compare the L_1 error and F1 score for respective parameters. Tab. 8 shows noticeable gains with 10 images and minor gains with 100 images. There is not much performance difference when evaluating on 100 or 500 images, which suggests that our framework is robust in generating consistent results when tested on different numbers of generated images by the same GM.

Content-independent fingerprint. Ideally our estimated fingerprint should be independent of the content of the image. That is, the fingerprint only includes the trace left by the GM while not

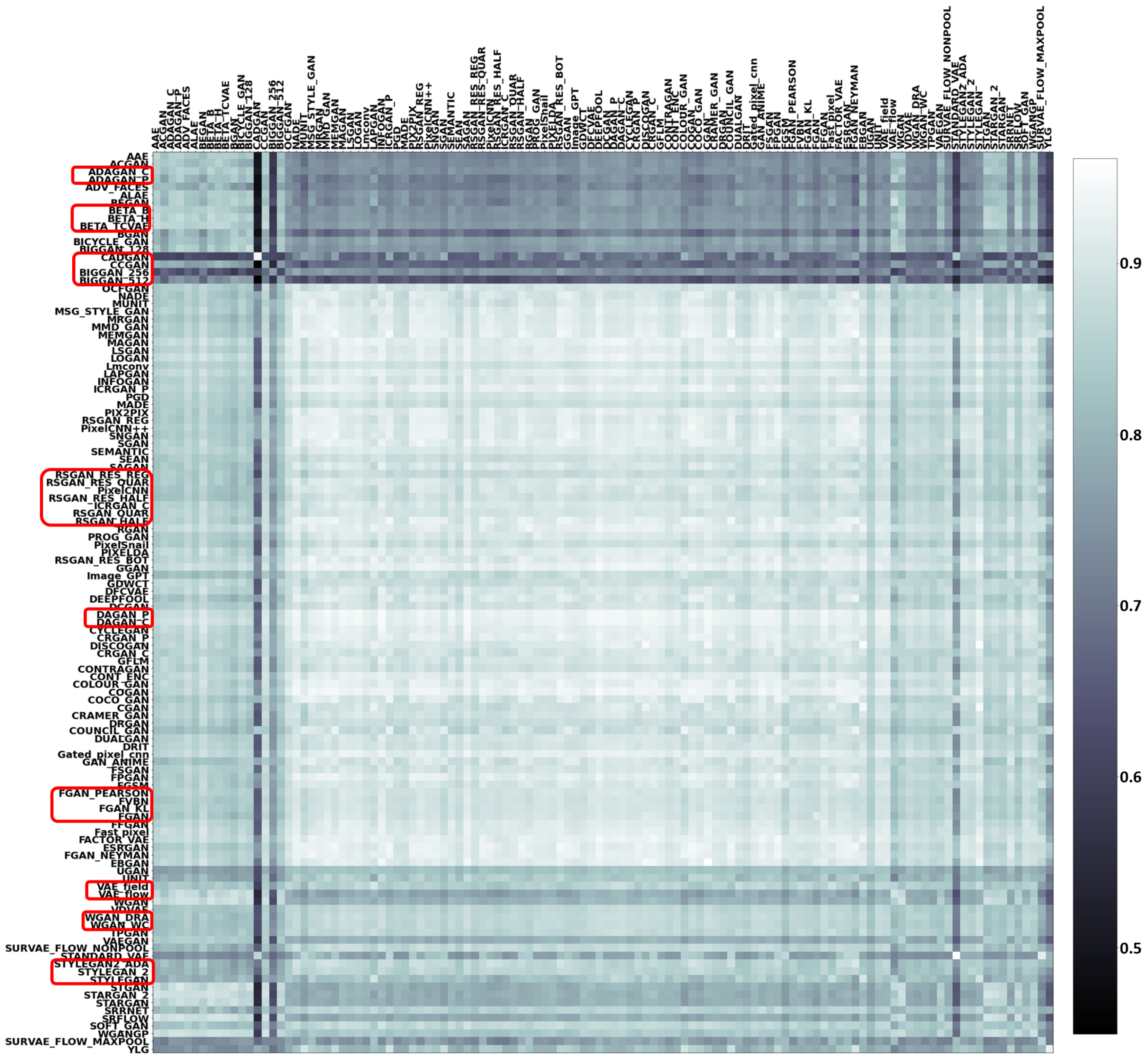


Fig. 11: Cosine similarity matrix for pairs of 116 GM's fingerprints. Each element of this matrix is the average Cosine similarities of 50 pairs of fingerprints from two GMs. We see the higher intra-GM and lower inter-GM similarities. We can also see GMs with similar network architecture or loss function are clustered together, as shown in the red boxes on the left.

indicating the content in any way. To validate this, we partition all GMs into four classes based on their contents: FACES (47 GMs), MNIST (25), CIFAR10 (31), and OTHER (13). Every class has images generated by the GMs belong to this class. We feed these images to a pre-trained FEN and obtain their fingerprints. Then we train a shallow network consisting of five convolutional layers and two fully connected layers for a 4-way classification. However, we observe the training cannot converge. This means that our estimated fingerprint from FEN doesn't have any content-specific properties for content classification. As a result, the model parsing of the hyperparameters doesn't leverage the content information across different GMs, which is a desirable property.

Evaluation on diffusion models. Due to the recent advancement of diffusion models for fake media generation, we evaluate our approach for these generative models. Specifically, we collect 7 diffusion models with $1K$ images each. We create 4 different test set splits, each set containing 3 diffusion models selected randomly. The remaining diffusion models, along with the full dataset is used for training. The result for our approach along with all the baselines is shown in Tab. 9. Our method clearly outperforms all the baselines, indicating the effectiveness of our approach for unseen models proposed in future. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

TABLE 9: Evaluation on diffusion models. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

Method	Network architecture		Loss function
	Continuous type L_1 error ↓	Discrete type F1 score ↑	F1 score ↑
Random ground-truth	$0.240 \pm 0.065/0.069$	0.664 ± 0.105	0.619 ± 0.083
No fingerprint	$0.211 \pm 0.080/0.078$	0.764 ± 0.112	0.711 ± 0.085
Using one parser	$0.201 \pm 0.045/0.041$	0.564 ± 0.101	0.654 ± 0.054
Ours	$\mathbf{0.189 \pm 0.051/0.049}$	$\mathbf{0.787 \pm 0.099}$	$\mathbf{0.724 \pm 0.076}$

TABLE 10: Binary classification performance for coordinated misinformation attack.

Method	AUC (%)	Classification accuracy (%)
FEN	83.5	76.85
FEN + PN	**87.3**	**80.6**

4.4 Visualization

Figure 10 shows an estimated fingerprint image and its frequency spectrum averaged over 25 randomly selected images per GM. We observe that estimated fingerprints have the desired properties defined by our loss terms, including low magnitude and highlights in middle and high frequencies.

We also find that the fingerprints estimated from different generated images of the same GM are similar. To quantify this, we compute a Cosine similarity matrix $\mathbf{C} \in \mathbb{R}^{116\times116}$ where $\mathbf{C}(i,j)$ is the averaged Cosine similarity of 25 randomly sampled fingerprint pairs from GM i and j. The matrix $\mathbf{C}$ in Figure 11 clearly illustrates the higher intra-GM ad lower inter-GM fingerprint similarities.

4.5 Applications

Coordinated misinformation attack. Our model parsing framework can be leveraged to estimate whether there exists a coordinated misinformation attack. That is, given two fake images, we hope to classify whether they are generated from the same GM or not. We do so by computing the Cosine similarity between the hyperparameters parsed from the given two images. First, we train our framework on 101 GMs, and test on 15 seen GMs and 15 unseen GMs. The list of GMs are mentioned in the supplementary. To evaluate this task, we report the Area Under Curve (AUC) and the classification accuracy at the optimum threshold. The results are shown in Tab. 10 comparing two methods, just using FEN network and using both FEN and PN. We conclude that our framework using FEN and PN can identify whether two images came from the same source with around 80% accuracy. Using only FEN network to compare the similarities of the fingerprint performs worse. This justifies the benefit of using parsed parameters for coordinated misinformation attack.

In fact, due to the nature of our test set, each pair of test samples can come from five different categories, namely, 1. Same seen GM, 2. Same unseen GM, 3. Different seen GMs, 4. Different unseen GMs, and 5. One seen and one unseen GM. We show an analysis of the wrongly classified samples in Figure 12 with respect to total number of samples and total number of samples in each category. Around 70% of the wrongly classified samples belong to the category of images coming from categories having atleast one GM unseen in training which is expected. However, if one of the test GM was seen in training, the number of wrongly classified samples decreased. This can be advantageous in detecting a manipulated image from an unknown GM.

TABLE 11: AUC for deepfake detection on the Celeb-DF dataset [34].

Method	Training Data	AUC (%)
Methods training with *pixel-level* supervision		
Xception+Reg [15]	DFFD	64.4
Xception+Reg [15]	DFFD, UADFV	71.2
Methods training with *image-level* supervision		
Two-stream [59]	Private	53.8
Meso4 [60]		54.8
VA-LogReg [61]		55.1
DSP-FWA [62]		64.6
Multi-task [63]	FF	54.3
Capsule [64]	FF++	57.5
Xception-c40 [11]		65.5
Two-branch [25]		73.4
SPSL [26]		**76.8**
SPSL [26] (reproduced)		73.2
Ours (fingerprint)		69.6
Ours (image+fingerprint)		71.1
Ours (image+fingerprint+phase)		74.6
Ours (model parsing)		64.3
HeadPose [65]	UADFV	54.6
FWA [66]		56.9
Xception [15]		52.2
Xception+Reg [15]		57.1
Ours		**64.7**
Xception [15]	DFFD	63.9
Ours		**65.3**
Xception [15]	DFFD, UADFV	67.6
Ours		**70.2**

TABLE 12: Classification rates of image attribution. The baseline results are cited from [17].

Method	CelebA	LSUN
kNN	28.00	36.30
Eigenface [67]	53.28	-
PRNU [22]	86.61	67.84
Yu *et al.* [17]	99.43	98.58
Ours	**99.66**	**99.84**

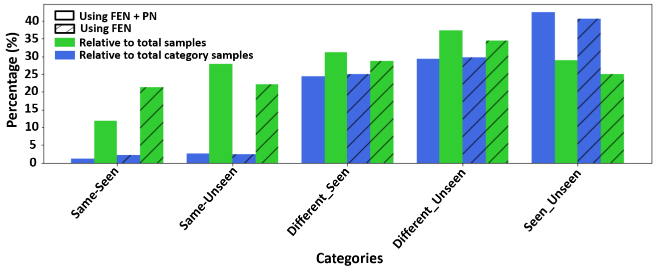


Fig. 12: Percentage of wrongly classified samples for five different categories of test sample pair. A larger number of sample pairs are wrongly classified if the pair of images come from same unseen GMs.

Deepfake detection. Our FEN can be adopted for deepfake detection by adding a shallow network for binary classification. We evaluate our method on the recently introduced Celeb-DF dataset [34]. We experiment with three training sets, UADFV, DFFD, and FF++, in order to compare with previous results. We follow the same training protocols used in [15] for UADFV and DFFD and [26] for FF++.

We report the AUC in Tab. 11. Compared with methods trained on UADFV, our approach achieves a significantly better result, despite the more advanced backbones used by others. Our results when trained on DFFD and UADFV fall only slightly behind the best performance reported by Xception+Reg [15]. Importantly, however, they trained with pixel-level supervision which is typically unavailable. These results are provided for completeness, but are not directly comparable to all other methods trained with only image-level supervision for binary classification. Compared to all other methods, our method achieves the highest deepfake detection AUC.

Finally, we compare the performance of our method when trained on FF++ dataset. [26] performs the best by using the phase information as an additional channel to the Xception classifier. However, as the pre-trained models were not released for [26], we reproduce their method and report the performance shown in Tab. 11. We observe a performance gap between the reproduced and reported performance which should be further investigated in the future. Following [26], we concatenate the fingerprint information with the RGB image and phase channels which are passed through a Xception classifier. Our method outperforms the reproduced performance of [26] showing the additional benefit of our fingerprint. Finally, we also perform the classification based on the pre-trained model parsing network and fine-tune it using the classification loss. The performance deteriorated compared to using the fingerprint. This shows that although the model parsing network have some deepfake detection abilities, they are less informative to perform deepfake detection well.

Image attribution. Similar to deepfake detection, we use a shallow network for image attribution. The only difference is that image attribution is a multi-class task and depends on the number of GMs during training. Following [17], we train our model on $100K$ genuine and $100K$ fake face images each from four GMs: SNGAN [68], MMDGAN [69], CRAMERGAN [70] and ProGAN [4], for five-class classification. Tab. 12 reports the performance. Our result on CelebA [34] and LSUN [54] outperform the performance in [17]. This again validates the generalization ability of the proposed fingerprint estimation.

5 CONCLUSION

In this paper, we define the model parsing problem as inferring the network architectures and training loss functions of a GM from the generative images. We make the first attempt to tackle this challenging problem. The main idea is to estimate the fingerprint for each image and use it for model parsing. Four constraints are developed for fingerprint estimation. We propose hierarchical learning to parse the hyperparameters in coarse-level and fine-level that can leverage the similarities between different GMs. Our fingerprint estimation framework can not only perform model parsing, but also extend to detecting coordinated misinformation attack, deepfake detection and image attribution. We have collected a large-scale fake image dataset from 116 different GMs. Various experiments have validated the effects of different components in our approach.

ACKNOWLEDGEMENT

This work was partially supported by Facebook AI. This material, except Section 4.5 and related efforts, is based upon work partially supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112090131 to Xiaoming Liu at Michigan State University.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014. 1

[2] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019. 1, 4

[3] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *CVPR*, 2018. 1

[4] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *ICLR*, 2018. 1, 4, 14

[5] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014. 1

[6] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-VAE," in *NeurIPS*, 2017. 1

[7] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *NeurIPS*, 2018. 1

[8] P. Dhariwal and A. Q. Nichol, "Diffusion models beat GANs on image synthesis," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=AAWuCvzaVt 1

[9] C. Waldemarsson, *Disinformation, Deepfakes & Democracy; The European response to election interference in the digital age*. The Alliance of Democracies Foundation, 2020. 1

[10] V. Heath, "From a sleazy Reddit post to a national security threat: A closer look at the deepfake discourse," in *Disinformation and Digital Democracies in the 21st Century*. The NATO Association of Canada, 2019. 1

[11] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to detect manipulated facial images," in *ICCV*, 2019. 1, 2, 3, 13

[12] S. McCloskey and M. Albright, "Detecting GAN-generated imagery using saturation cues," in *ICIP*, 2019. 1, 2, 3, 5

[13] L. Guarnera, O. Giudice, and S. Battiato, "Deepfake detection by analyzing convolutional traces," in *CVPRW*, 2020. 1, 2, 3, 5

[14] F. Marra, C. Saltori, G. Boato, and L. Verdoliva, "Incremental learning for the detection and classification of GAN-generated images," in *WIFS*, 2019. 1, 2, 3

[15] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain, "On the detection of digital face manipulation," in *CVPR*, 2020. 1, 2, 3, 13

[16] Y. Nirkin, L. Wolf, Y. Keller, and T. Hassner, "Deepfake detection based on the discrepancy between the face and its context," *arXiv preprint arXiv:2008.12262*, 2020. 1, 2, 3

[17] N. Yu, L. S. Davis, and M. Fritz, "Attributing fake images to GANs: Learning and analyzing GAN fingerprints," in *ICCV*, 2019. 1, 2, 3, 4, 5, 7, 13, 14

[18] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIXSS*, 2016. 2, 3

[19] S. J. Oh, M. Augustin, M. Fritz, and B. Schiele, "Towards reverse-engineering black-box neural networks," in *ICLR*, 2018. 2, 3

[20] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *DAC*, 2018. 2, 3

[21] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *USENIXSS*, 2019. 2, 3

[22] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, "Do GANs leave artificial fingerprints?" in *MIPR*, 2019. 2, 3, 4, 13

[23] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot... for now," in *CVPR*, 2020. 2, 3, 4, 5

[24] X. Zhang, S. Karaman, and S.-F. Chang, "Detecting and simulating artifacts in GAN fake images," in *WIFS*, 2019. 2, 3, 4, 5

[25] I. Masi, A. Killekar, R. M. Mascarenhas, S. P. Gurudatt, and W. AbdAlmageed, "Two-branch recurrent network for isolating deepfakes in videos," in *ECCV*. Springer, 2020. 2, 3, 13

[26] H. Liu, X. Li, W. Zhou, Y. Chen, Y. He, H. Xue, W. Zhang, and N. Yu, "Spatial-phase shallow learning: rethinking face forgery detection in frequency domain," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 772–781. 2, 3, 13, 14

[27] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006. 2, 3, 4

[28] M. Goljan, J. Fridrich, and T. Filler, "Large scale test of sensor fingerprint camera identification," *Media forensics and security*, vol. 7254, p. 72540I, 2009. 2

[29] K. Kurosawa, K. Kuroki, and N. Saitoh, "CCD fingerprint method-identification of a video camera from videotaped images," in *ICIP*, 1999. 2

[30] T. Filler, J. Fridrich, and M. Goljan, "Using sensor pattern noise for camera model identification," in *ICIP*, 2008. 2

[31] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, "Compressed fingerprint matching and camera identification via random projections," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1472–1485, 2015. 2

[32] J. Lukáš, J. Fridrich, and M. Goljan, "Detecting digital image forgeries using sensor pattern noise," *Security, Steganography, and Watermarking of Multimedia Contents VIII*, vol. 6072, p. 60720Y, 2006. 2

[33] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, "Determining image origin and integrity using sensor noise," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, 2008. 2

[34] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, "Celeb-DF: A large-scale challenging dataset for deepfake forensics," in *CVPR*, 2020. 3, 7, 13, 14

[35] D. Cozzolino and L. Verdoliva, "Noiseprint: a CNN-based camera model fingerprint," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 144–159, 2019. 3

[36] T. K. Moon, "The expectation-maximization algorithm," *Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996. 3

[37] L. Chai, D. Bau, S.-N. Lim, and P. Isola, "What makes fake images detectable? Understanding properties that generalize," in *ECCV*, 2020. 3

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017. 3

[39] Y. Nirkin, I. Masi, A. T. Tuan, T. Hassner, and G. Medioni, "On face segmentation, face swapping, and face perception," in *FGR*. IEEE, 2018, pp. 98–105. 3

[40] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys*, vol. 54, no. 2, 2021. 3

[41] A. Jabbar, X. Li, and B. Omar, "A survey on generative adversarial networks: Variants, applications, and training," *arXiv preprint arXiv:2006.05132*, 2020. 3

[42] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015. 4

[43] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012. 4

[44] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009. 4

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009. 4

[46] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017. 4

[47] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017. 4

[48] A. Jourabloo, Y. Liu, and X. Liu, "Face de-spoofing: Anti-spoofing via noise modeling," in *ECCV*, 2018. 4, 5

[49] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *CVPR*, 2019. 5

[50] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *ICML*, 2018. 5

[51] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018. 5

[52] P. Bholowalia and A. Kumar, "Ebk-means: A clustering technique based on elbow method and k-means in wsn," *International Journal of Computer Applications*, vol. 105, no. 9, 2014. 6

[53] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013. 6

[54] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015. 7, 14

[55] A. K. Srivastava, V. K. Srivastava, and A. Ullah, "The coefficient of determination and its adjusted version in linear regression models," *Econometric reviews*, vol. 14, no. 2, pp. 229–240, 1995. 8

[56] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. 8

[57] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement," *Association for Computing Machinery SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 49–57, 2010. 8

[58] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data–recommendations for the use of performance metrics," in *ACII*, 2013. 8

[59] X. Han, V. Morariu, P. I. Larry Davis *et al.*, "Two-stream neural networks for tampered face detection," in *CVPRW*, 2017. 13

[60] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: a compact facial video forgery detection network," in *WIFS*, 2018. 13

[61] F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," in *WACVW*, 2019. 13

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015. 13

[63] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, "Multi-task learning for detecting and segmenting manipulated facial images and videos," in *BTAS*, 2019. 13

[64] H. H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-forensics: Using capsule networks to detect forged images and videos," in *ICASSP*, 2019. 13

[65] X. Yang, Y. Li, and S. Lyu, "Exposing deep fakes using inconsistent head poses," in *ICASSP*, 2019. 13

[66] Y. Li and S. Lyu, "Exposing DeepFake videos by detecting face warping artifacts," in *CVPRW*, 2019. 13

[67] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America*, vol. 4, no. 3, pp. 519–524, 1987. 13

[68] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018. 14

[69] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "MMD GAN: Towards deeper understanding of moment matching network," in *NeurIPS*, 2017. 14

[70] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The cramer distance as a solution to biased wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017. 14

[71] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[72] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *SP*, 2017.

[73] B. Škrlj, S. Džeroski, N. Lavrač, and M. Petkovič, "Feature importance estimation with self-attention networks," in *ECAI*, 2019.

[74] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva, "A bayesian-MRF approach for PRNU-based image forgery detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 554–567, 2014.

[75] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques," in *ICIP*, 2014.

[76] S. Chakraborty and M. Kirchner, "PRNU-based image manipulation localization with discriminative random fields," *Electronic Imaging*, vol. 2017, no. 7, pp. 113–120, 2017.

[77] P. Korus and J. Huang, "Multi-scale analysis strategies in PRNU-based tampering localization," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 809–824, 2016.

[78] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[79] H. Kim and A. Mnih, "Disentangling by factorising," in *ICML*, 2018.

[80] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[81] C. H. Lin, C.-C. Chang, Y.-S. Chen, D.-C. Juan, W. Wei, and H.-T. Chen, "COCO-GAN: generation by parts via conditional coordinating," in *ICCV*, 2019.

[82] Y. Yu, Z. Gong, P. Zhong, and J. Shan, "Unsupervised representation learning with deep convolutional neural network for remote sensing images," in *ICIG*, 2017.

[83] X. Hou, L. Shen, K. Sun, and G. Qiu, "Deep feature consistent variational autoencoder," in *WACV*, 2017.

[84] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning GAN for pose-invariant face recognition," in *CVPR*, 2017.

[85] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, "Towards large-pose face frontalization in the wild," in *ICCV*, 2017.

[86] Y. Nirkin, Y. Keller, and T. Hassner, "FSGAN: Subject agnostic face swapping and reenactment," in *ICCV*, 2019.

[87] R. Wang, A. Cully, H. J. Chang, and Y. Demiris, "MAGAN: Margin adaptation for generative adversarial networks," *arXiv preprint arXiv:1704.03817*, 2017.

[88] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," in *ICLR*, 2017.

[89] A. F. Ansari, J. Scarlett, and H. Soh, "A characteristic function approach to deep implicit generative modeling," in *CVPR*, 2020.

[90] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *ICML*, 2019.

[91] P. Zhu, R. Abdal, Y. Qin, and P. Wonka, "SEAN: Image synthesis with semantic region-adaptive normalization," in *CVPR*, 2020.

[92] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "StarGAN v2: Diverse image synthesis for multiple domains," in *CVPR*, 2020.

[93] M. Liu, Y. Ding, M. Xia, X. Liu, E. Ding, W. Zuo, and S. Wen, "STGAN: A unified selective transfer network for arbitrary image attribute editing," in *CVPR*, 2019.

[94] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *CVPR*, 2020.

[95] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception GAN for photorealistic and identity preserving frontal view synthesis," in *ICCV*, 2017.

[96] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *ICML*, 2016.

[97] C. Chen, Z. Xiong, X. Liu, and F. Wu, "Camera trace erasing," in *CVPR*, 2020.

[98] L. Zhao, M. Zhang, H. Ding, and X. Cui, "Mff-net: Deepfake detection network based on multi-feature fusion," *Entropy*, vol. 23, no. 12, p. 1692, 2021.

Vishal Asnani is pursuing his Ph. D. degree in the Computer Science and Engineering department from Michigan State University since 2021. He received his Bachelor's degree in Electrical and Instrumentation Engineering from Birla Institute of technology and Science, Pilani, India in 2019. His research interests include computer vision and machine learning with a focus on the studying of generative models and deepfake detection.

Xi Yin is a Research Scientist at Facebook AI Applied Research team. She received her Ph.D. degree in Computer Science and Engineering from Michigan State University in 2018. Before joining Facebook AI, she was an Senior Applied Scientist at Microsoft Cloud and AI. Her research is focused on computer vision, deep learning, vision and language. She has co-authored 18 papers in top vision conferences and journals, and filed 3 U.S. patents. She has received Best Student Paper Award at WACV 2014. She is an Area Chair for IJCB 2021 and ICCV 2021.

Tal Hassner Tal Hassner received his M.Sc. and Ph.D. degrees in applied mathematics and computer science from the Weizmann Institute of Science in 2002 and 2006, respectively. In 2008 he joined the Department of Math. and Computer Science at The Open Univ. of Israel where he was an Associate Professor until 2018. From 2015 to 2018, he was a senior computer scientist at the Information Sciences Institute (ISI) and a Visiting Research Associate Professor at the Institute for Robotics and Intelligent Systems, Viterbi School of Engineering, both at USC, CA, USA. From 2018 to 2019, he was a principal applied scientist at AWS Rekognition. Since 2019 he is a research manager at Meta (formally Facebook). He served as a program chair at WACV'18, ICCV'21, and ECCV'22, a general chair for WACV'24, a workshop chair at CVPER'20, tutorial chair at ICCV'17, and area chair in CVPR, ECCV, AAAI, and others. Finally, he is an associate editor at IEEE-TPAMI and IEEE-TBIOM.

Xiaoming Liu is a MSU Foundation Professor at the Department of Computer Science and Engineering of Michigan State University. He received the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University in 2004. Before joining MSU in Fall 2012, he was a research scientist at General Electric (GE) Global Research. His research interests include computer vision, machine learning, and biometrics. As a co-author, he is a recipient of Best Industry Related Paper Award runner-up at ICPR 2014, Best Student Paper Award at WACV 2012 and 2014, Best Poster Award at BMVC 2015, and Michigan State University College of Engineering Withrow Endowed Distinguished Scholar Award. He has been the Area Chair for numerous conferences, including CVPR, ICCV, ECCV, ICLR, NeurIPS, the Program CO-Chair of WACV'18, BTAS'18, AVSS'22 conferences, and General Co-Chair of FG'23 conference. He is an Associate Editor of Pattern Recognition Letters, Pattern Recognition, and IEEE Transactions on Image Processing. He has authored more than 150 scientific publications, and has filed 29 U.S. patents. He is a fellow of IAPR.